Assignment 4

Sriyaank Vadlamani

2023-04-30

```
library(rpart)
library(geosphere)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```
library(rpart.plot)
library(stringr)
library(moderndive)
library(mlr)
```

```
## Loading required package: ParamHelpers

## Warning message: 'mlr' is in 'maintenance-only' mode since July 2019.
## Future development will only happen in 'mlr3'
## (<https://mlr3.mlr-org.com>). Due to the focus on 'mlr3' there might be
## uncaught bugs meanwhile in {mlr} - please consider switching.
```

```
library(Metrics)
library(vip)
```

```
##
## Attaching package: 'vip'

## The following object is masked from 'package:utils':
##
##     vi
```

```
library(ggplot2)
```

```
set.seed(43023)
```

Datasets

```
train <- read.csv("train_data.csv")
test <- read.csv("test_data.csv")
```

```
train <- na.omit(train)
# test <- na.omit(test)
```

Adding Distance from JDF and Distance from Broadway columns

```
jfk <- matrix(c( -73.7781, 40.6413), nrow=1) # uses latitude and longitude of JFK airport
broadway <- matrix(c(-73.9747, 40.7908), nrow=1) # uses latitude and longitude of broadway

# The distances are divided by 1000 to avoid scientific notation on decision tree
train$distance_jfk <- distGeo(jfk, matrix(c(train$longitude, train$latitude), ncol=2)) / 1000
test$distance_jfk <- distGeo(jfk, matrix(c(test$longitude, test$latitude), ncol=2)) / 1000

train$distance_broadway <- distGeo(broadway, matrix(c(train$longitude, train$latitude), ncol=2)) / 1000
test$distance_broadway <- distGeo(broadway, matrix(c(test$longitude, test$latitude), ncol=2))/ 1000
```

Adding has_crime column

```
crime_data <- read.csv("NYC_Crime_Statistics.csv")
crime_dict <- with(crime_data, setNames(Zip.Codes, TOTAL.SEVEN.MAJOR.FELONY.OFFENSES))

train$zipcode <- str_sub(train$Location, -20, -16)
train$zipcode <- as.integer(train$zipcode)
```

```
## Warning: NAs introduced by coercion
```

```
train$has_crime <- ifelse(any(crime_data == train$zipcode), TRUE, FALSE)

train$has_crime <- train$zipcode %in% crime_data$Zip.Codes

test$zipcode <- str_sub(test$Location, -20, -16)
test$zipcode <- as.integer(test$zipcode)
```

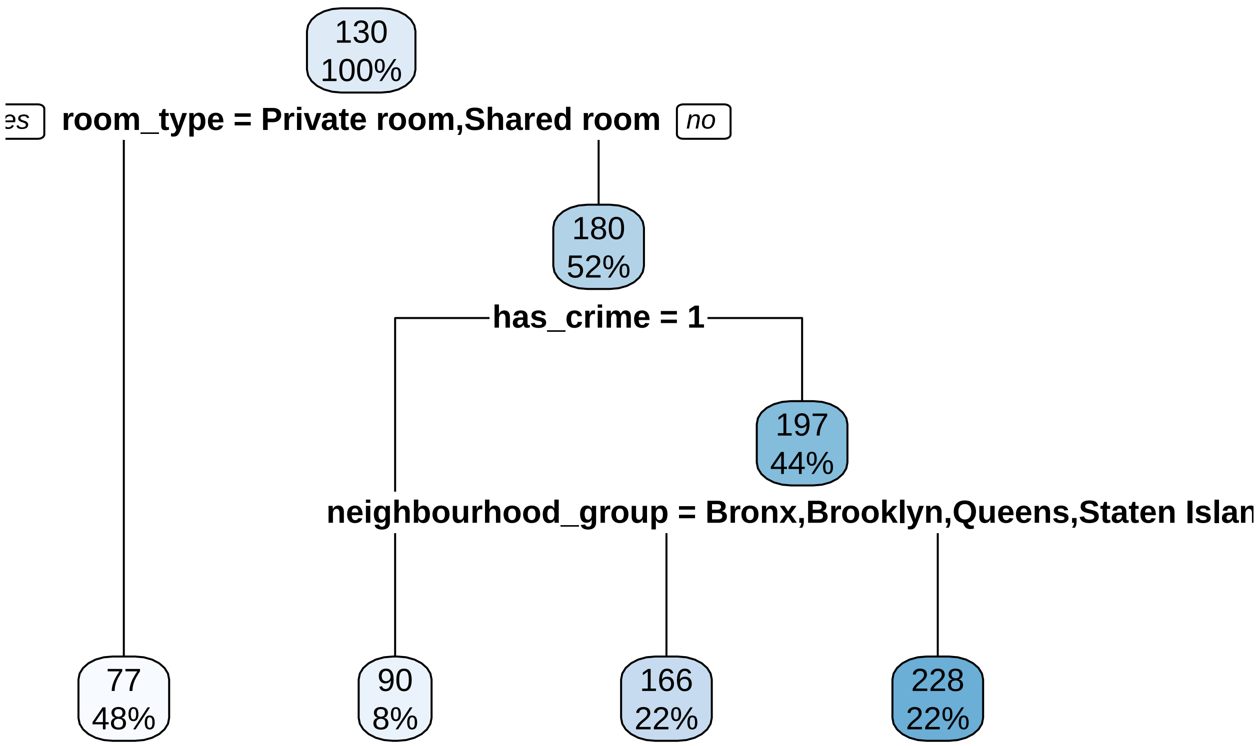
```
## Warning: NAs introduced by coercion
```

```
test$has_crime <- ifelse(any(crime_data == test$zipcode), TRUE, FALSE)

test$has_crime <- test$zipcode %in% crime_data$Zip.Codes
```

Test #1: Initial Test

```
train1 <- train %>% select(neighbourhood_group, room_type, distance_jfk, distance_broadway, has_crime, p
tree1 <- rpart(price ~ ., data = train1, method = "anova")
rpart.plot(tree1)
```



Accuracy Test

```r
predicted_values <- predict(tree1, train)
mae(predicted_values, train$price)
```

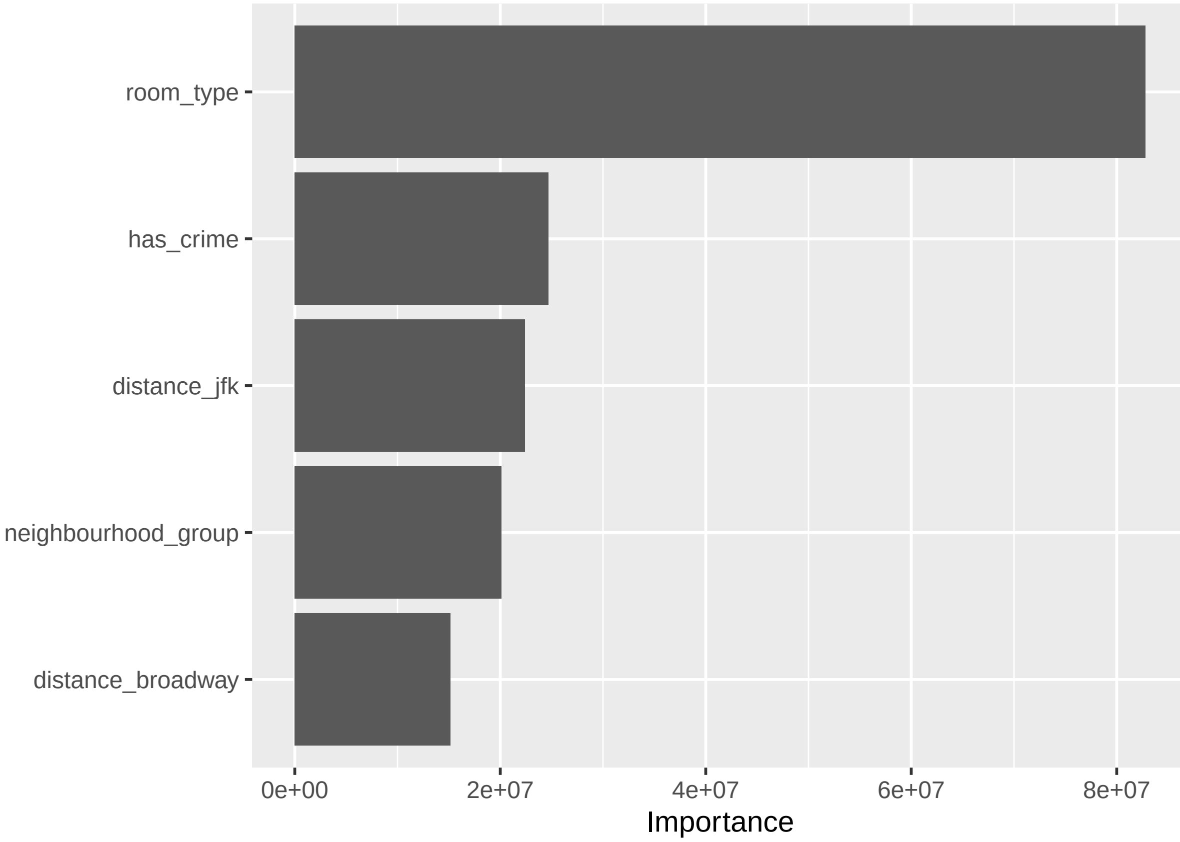
```
## [1] 58.65155
```

```r
rmse(predicted_values, train$price)
```

```
## [1] 182.5883
```

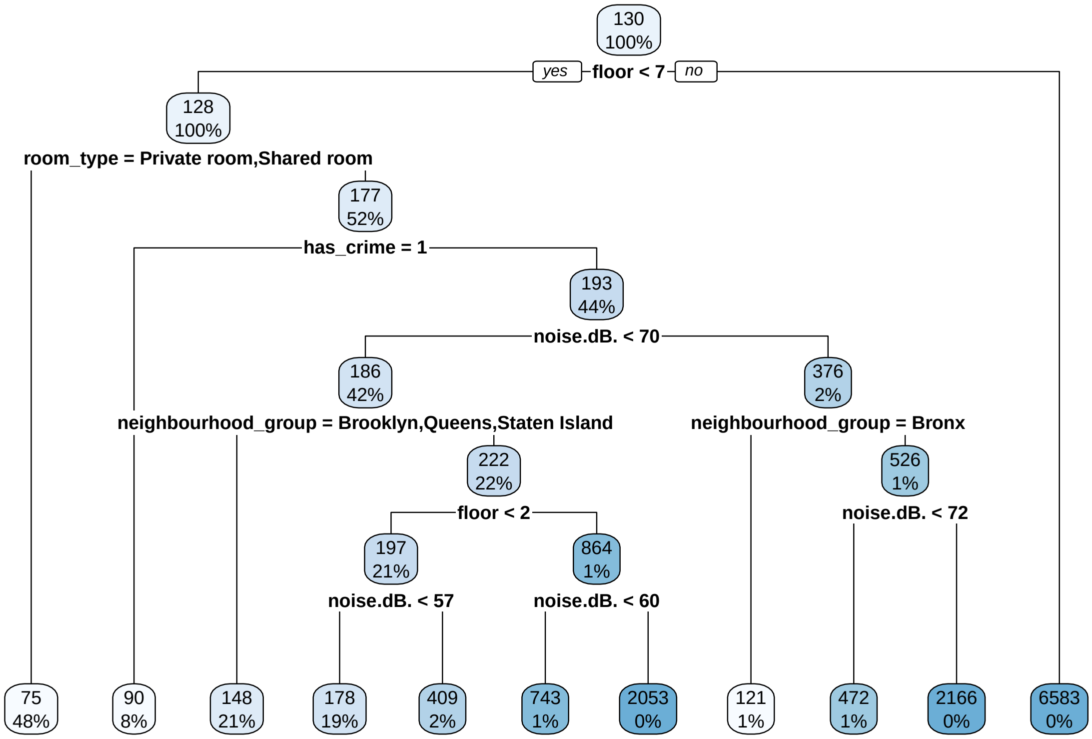
Feature Importance

```r
vip(tree1)
```



Test #2: More variables

```
train2 <- train %>% select(neighbourhood_group, floor, room_type, distance_jfk, distance_broadway, minim
tree2 <- rpart(price ~ ., data = train2, method = "anova")
rpart.plot(tree2)
```



Accuracy Test

```r
predicted_values <- predict(tree2, train)
mae(predicted_values, train$price)
```

```
## [1] 43.91276
```

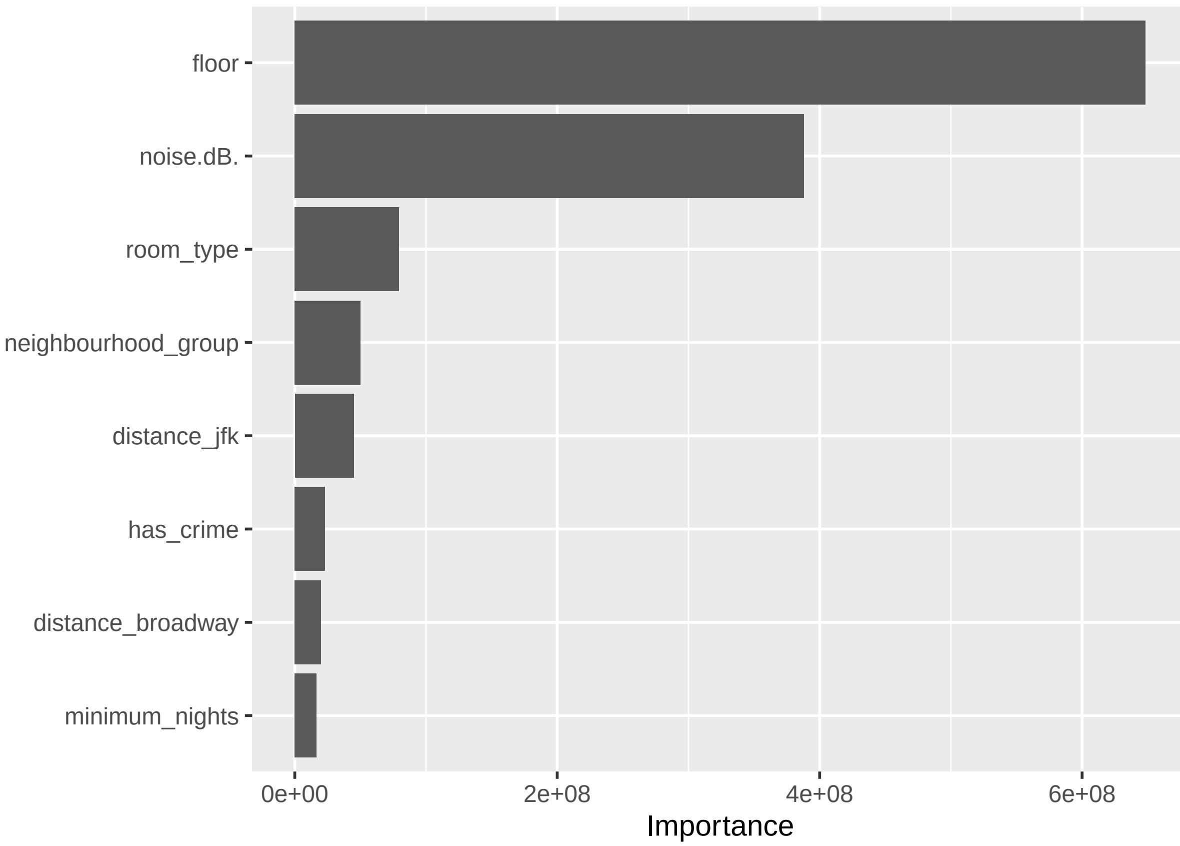
```r
rmse(predicted_values, train$price)
```

```
## [1] 105.5126
```

Better than before, but maybe hyperparameter tuning the model to find optimal max depth can improve it

Feature Importance

```r
vip(tree2)
```



Hyperparameter tuning

Parameter Set

```
getParamSet("regr.rpart")
```

```
##                    Type len  Def   Constr Req Tunable Trafo
## minsplit         integer   -   20 1 to Inf   -    TRUE     -
## minbucket        integer   -    - 1 to Inf   -    TRUE     -
## cp               numeric   - 0.01   0 to 1   -    TRUE     -
## maxcompete       integer   -    4 0 to Inf   -    TRUE     -
## maxsurrogate     integer   -    5 0 to Inf   -    TRUE     -
## usesurrogate    discrete   -    2    0,1,2   -    TRUE     -
## surrogatestyle discrete   -    0      0,1   -    TRUE     -
## maxdepth         integer   -   30  1 to 30   -    TRUE     -
## xval             integer   -   10 0 to Inf   -   FALSE     -
```

Make parameter sets

```
train3 <- train2
train3[sapply(train2, is.character)] <- lapply(train3[sapply(train2, is.character)], as.factor)
train3[sapply(train2, is.logical)] <- lapply(train3[sapply(train2, is.logical)], as.factor)

tree_params <- makeRegrTask(data=train3, target="price")

param_grid <- makeParamSet(makeDiscreteParam("maxdepth", values=1:30), makeNumericParam("cp", lower = 0
```

Define Grid

```
control_grid = makeTuneControlGrid()
```

Define Cross Validation

```
resample = makeResampleDesc("CV", iters = 3L)
```

Define Measure

```
measure = list(mlr::mae, mlr::rmse)
```

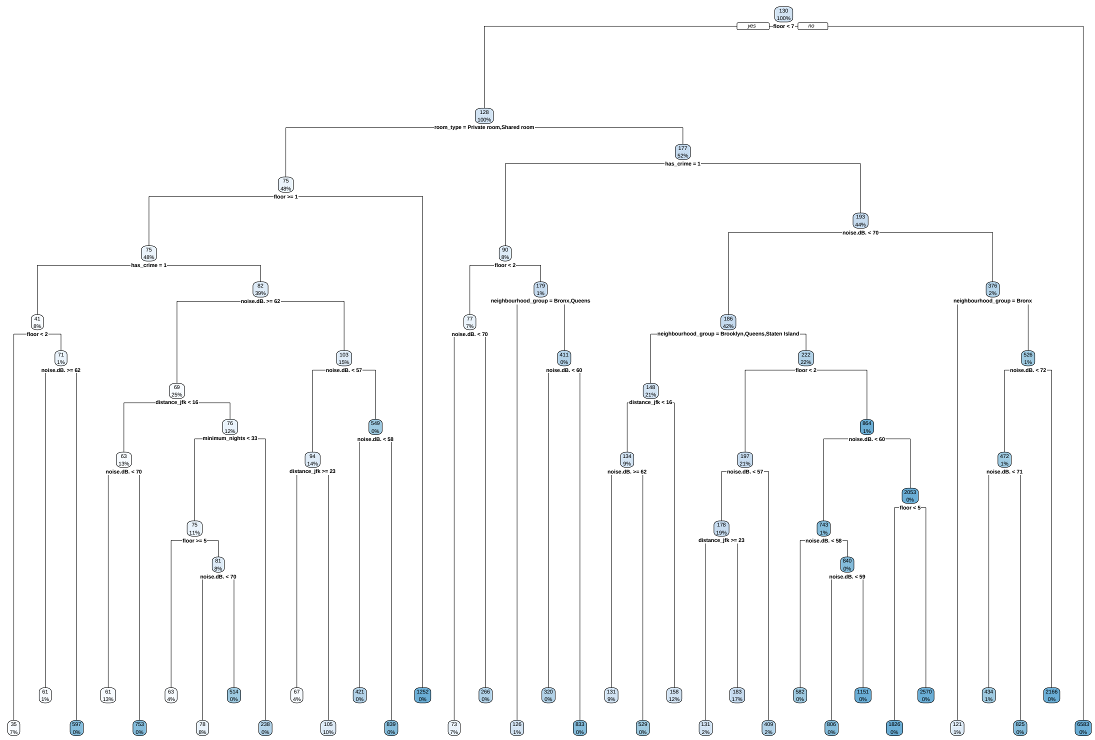
tuneParameters

```
# tuneparam <- tuneParams(learner='regr.rpart',
#  task=tree_params,
#  resampling = resample,
#  measures = measure,
#  par.set=param_grid,
#  control=control_grid,
#  show.info = TRUE)
```

Hyperparameter tuning commented out so the pdf doesn't give several pages of results, but it listed depth 24 as best depth for mae and rmse and 0.001 as the optimal complexity parameter. Let's test it

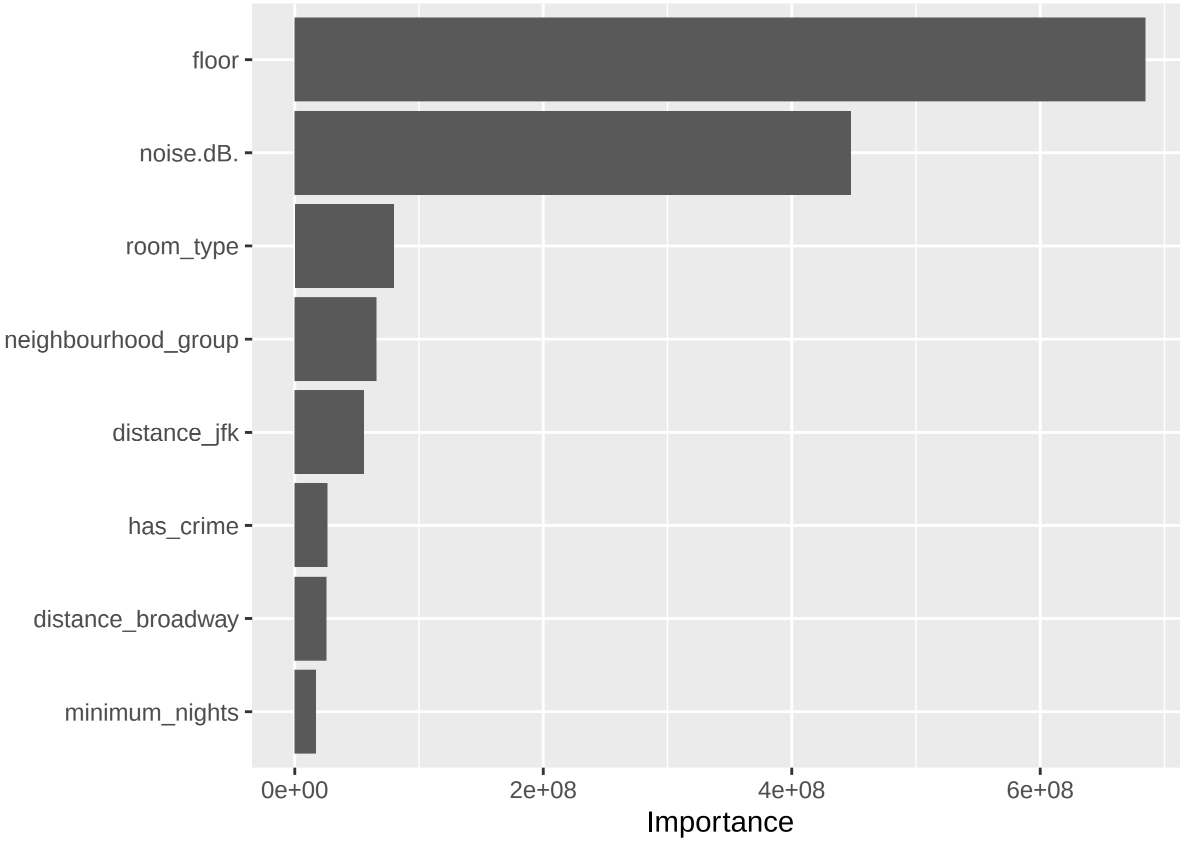
Test 3: Hyperparameter tuned test

```
tree3 <- rpart(price ~ ., data = train2, method = "anova", control = c(maxdepth = 24, cp = 0.001))
rpart.plot(tree3)
```



Feature Importance

```
vip(tree3)
```



Accuracy test

```
predicted_values <- predict(tree3, train2)
mae(predicted_values, train2$price)
```

```
## [1] 36.35329
```

```
rmse(predicted_values, train2$price)
```

```
## [1] 92.0964
```

This is the same tree as before. Let's use it

Predict the test data

Prediction time!

```
test$price <- predict(tree3, test, na.action = na.exclude)
write.csv(test, "test_final.csv")

submission <- test %>% select("id", "price")

write.csv(submission, "Spring23_ds_capstone_submission.csv", row.names=FALSE)
```